intel

APPLICATION NOTE

AP-114

October 1981

Using the iSBC® 88/40 Measurement and Control Computer in PID Applications

Peter Andersen
OMS Applications Engineering

Order Number: 210263-001

INTRODUCTION

During the past twenty years, the automated process control industry has matured significantly. This is due to the introduction of the digital computer as an element of the control system. At the beginning of this period, the use of the digital computer was limited to a supervisory status in which the actual control was performed by various combinations of relay, analog, and pneumatic systems. Today, systems are off-the-shelf digital hardware and software to perform all the control applications. Indeed, the use of the hardware/software combination has opened entirely new areas of control applications.

The significant increase in computer capabilities and the corresponding reduction in size has been accompanied by a substantial drop in cost. This has led to a strong incentive for users to employ computers in totally new application areas which have resulted from this change in economics. Twenty years ago, few computer control projects were initiated and those which were could only be justified economically in terms of control systems which controlled upwards of 100 loops. Today, a microcomputer system can be justified for a small process which contains as few as 3 or 4 control loops.

Today, the control system engineer's decision is not so much an economic justification of a digital process as it is a choice of whether to use a single or a multiple microcomputer based design.

The trend toward the use of digital technology in the control world has been driven, in part, by the products which have been introduced into the marketplace by Intel Corporation. A recently announced product, the iSBC 88/40 Measurement and Control Computer, is intended to further simplify the implementation of digital technologies into varied control applications and is the subject of this application note. Its architecture is well suited for both single microcomputer and multicomputing environments. The board is also easily adapted to a wide variety of input/output configurations through on-board facilities and iSBX MULTIMODULE expansion boards.

Generalized Computer Application Areas

Those applications in which computers are finding acceptance can generally be broken down into two broad areas. The first involves the acquisition and manipulation of process data by the computer, and is sometimes referred to as being a class of passive applications. The second, known as active systems, also involves the manipulation of the process itself. The systems in the latter class also provide various degrees of passive data manipulation.

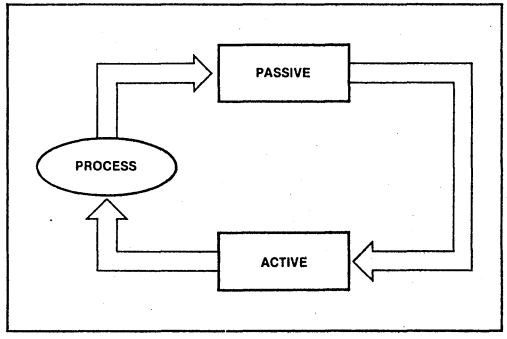

Figure 2. Classes of Computer Applications

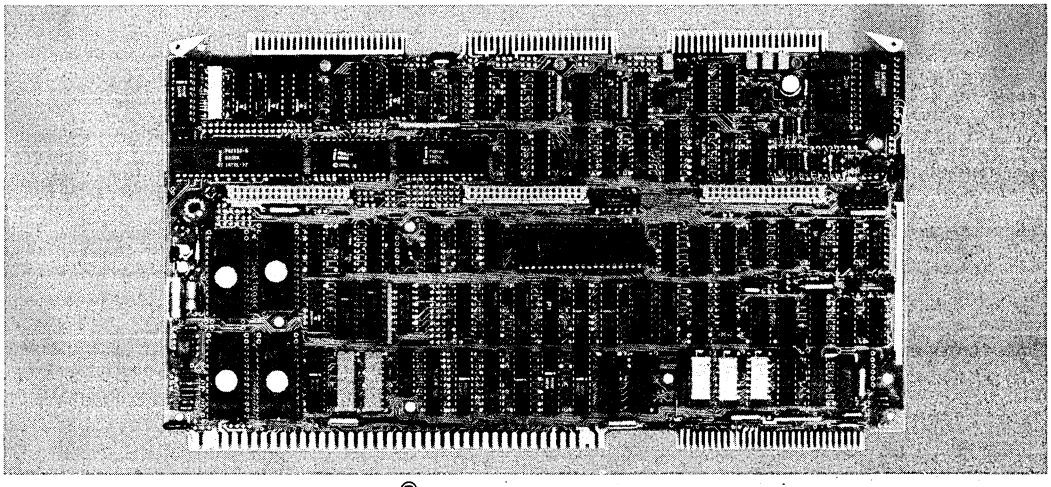

Figure 1. The iSBC® 88/40 Measurement and Control Computer

At first glance, the area of passive computer applications seems to have little or nothing to do with process control; however, many computer design projects are being split into two phases. One phase is to characterize the process and the second is concerned with the actual control system. Many designs never move from phase one and are used as data acquisition systems.

The majority of passive systems involve measuring physical parameters of the process application. Examples are the measurement of pressure, temperature, flow, force, and level. Most transducers associated with these physical parameters provide an analog signal which is proportional to the physical property being sensed. Thus, the ability to measure analog voltages is a requirement of process control systems, both active and passive.

The iSBC 88/40 Measurement and Control Computer is ideally suited for these classes of systems because of the board's built-in analog to digital conversion circuitry. Each input channel (there are 16 differential or 32 single-ended channels on the board) has its own programmable gain which can be software selected to provide full scale inputs ranging from 20 millivolts to 10 volts. The board is thus compatible with most commonly available transducer elements. Examples of typical interface drivers are given in later sections of this application note.

Active applications must interact with the control system in order to manage the process. This normally involves the activation and movement of a mechanical element which is incorporated into the process loop. An amplifier and transducer are required to convert the electrical output of the controller into mechanical energy. The majority of these activators are electro-pneumatic, requiring both an electrical control signal (usually 4-20 milliamps) from the controller and an air supply for its internal pneumatic amplifier. Less common, but still in substantial numbers, are activators which use either a frequency input control signal or stepping motors.

Again, the iSBC 88/40 Measurement and Control Computer provides features designed to allow easy interface to various control actuators. For those actuators using digital frequencies or stepping motors, the board has a parallel output capability to drive up to 24 digital lines. Pulse output signals can be routed from programmable timers/counters (to generate a variety of pulse type outputs) to the external I/O devices. Analog actuators can be driven using the iSBX 328 Analog Output MULTIMODULE Board. This board connects to the measurement and control computer using one of the three iSBX connectors on the iSBC 88/40 board. Each iSBX 328 board can generate up to 8 analog output signals, each of which can function in either a voltage or current (4-20 milliamps) output mode.

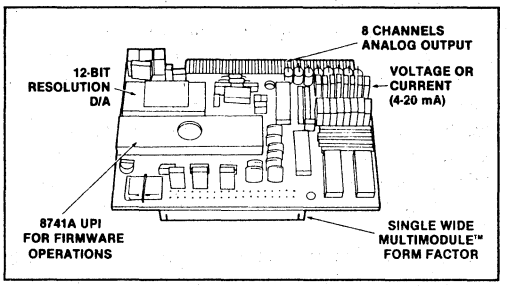

Figure 3. iSBX™ 328 Analog Output MULTIMODULE™ Board

Computer Processing Capabilities

The key to the rapid growth of digital computers in process control has been the flexibility offered by the software. The same hardware can be used in widely varying applications by allowing customization through software programmming. To be successful in the process control marketplace, a digital computer system must be designed in a manner which optimizes the hardware/software relationships. The iSBC 88/40 Measurement and Control Computer does this well.

A powerful instruction set is mandatory if operations are to be efficiently performed by the processor. An instruction set optimized to perform business operations will perform poorly in an industrial process application. The processor used on the iSBC 88/40 board is the Intel iAPX 88/10 microprocessor. This third generation microprocesor is suitable for a wide spectrum of applications. The large application domain is made possible by the processor's dual operating modes and built-in multiprocessing features. The iAPX 88/10 microprocessor is from four to six times more powerful than the 8080A microprocessor.

The high performance of the iAPX 88/10 microprocessor is realized by combining an internal data path with a pipelined architecture that allows instructions to be prefetched during spare bus cycles. Also contributing to performance is a compact instruction format that enables more instructions to be fetched in a given amount of time.

Software for high-performance iAPX 88/10 processors need not be written in assembly language (although it certainly can be). The CPU is designed to provide direct hardware support for programs written in high level languages such as Intel's PL/M-86. Because most high level languages store variables in memory, the instruc-

tion set supports direct operation on memory operands, including operands on the stack. The hardware addressing modes provide efficient implementations of based variables, arrays, arrays of structures and other high level language data constructs. Hardware multiplication and division of signed and unsigned binary numbers, as well as unpacked decimal numbers, is fully supported by the CPU. In all, about 300 forms of machine level instructions are supported by the iAPX 88/10 processor.

Memory Options

A key design requirement for the iSBC 88/40 Measurement and Control Computer was to have the board support a variety of memory types and capacities. The result is a product which can easily be configured to meet a wide range of process control application requirements.

Program storage support for small to very large applications is obtained through the board's ability to include EPROM storage capacities ranging up to 64K bytes. Maximum standard storage capacity is from 8K bytes (using 2716 EPROM devices) to 32K bytes (using the 2764 EPROM). An optional EPROM expansion MULTIMODULE board can be mounted onto the iSBC 88/40 board to double the memory storage capacities.

Variables used in an application are usually stored in RAM memory. A standard on-board RAM capacitiy of 4K bytes is included on the measurement and control computer. In order to efficiently support multicomputer system design, 1K bytes of this memory is dual-ported. Dual-porting introduces a three bus system architecture to system design. An on-board local bus creates a data path between the iAPX 88/10 CPU and its local RAM. Data paths to RAM located on other iSBC boards are provided by the facilities of the MULTIBUS system bus. Finally, a third bus provides a gateway into the local RAM by other MULTIBUS single board computers or bus masters. If additional RAM is required, a small MULTIMODULE RAM expansion board can be attached to the iSBC 88/40 board to add 4K bytes of random access memory.

Even more flexibility can be gained by using unneeded EPROM memory sockets. Because JEDEC standard 24/28 pin sockets have been used, byte wide RAM modules can be inserted into areas of the EPROM memory space. The use of this RAM can considerably enhance the design of certain applications. The board capabilities are such that it is not necessary to have all devices residing in the EPROM sockets be of the same type or size.

Many process control applications require the use of non-volatile memory for the storage of parameter lists and system setpoints. Provision has been made on the iSBC 88/40 Measurement and Control Computer to fully support Intel's new 2816 Electrically Eraseable and Programmable Read Only Memory (E^2PROM). This device gives the user 2K bytes of memory. Depending on the application, up to eight devices (16K bytes) can be used on the iSBC 88/40 board. The board includes all required voltages and wave-shaping circuits to fully support the use of the 2816. A byte of 2816 memory can be programmed in 16 milliseconds. A subsequent section of this application note contains a comprehensive discussion of the operation of the board with the 2816.

Arithmetic Functions

Using computers as an element in a control system leads to extensive arithmetic and mathematical functions. To be effective and attractive to the designer, a computer board must provide a wide range of mathematical capabilities. The iSBC 88/40 Measurement and Control Computer easily meets these needs with varying capabilities for hardware and software functions.

Many applications are adequately handled using the hardware add/subtract and multiply/divide instructions of the on-board iAPX 88/10 processor. Functions needing integer arithmetic of varying precisions are easily programmed using this facility. In some cases, more complex operations may require the use of software libraries to gain the required mathematical functions. The speed and instruction set of the CPU, in conjunction with PL/M-86 statements, make programmers comfortable with these operations.

As processes become more involved and their control algorithms more complex, the need for the processor to support more precise numbers becomes important. The additional precision is usually obtained through the use of a floating point representation. Intel supplies several tools which simplify the implementation of systems requiring floating point operations. Complete support for the floating point numbers is provided as an integral part of the PL/M-86 compiler. Thus, variables can be specified as real numbers. The compiler will perform all numerical operations on these numbers in the floating point format. The data formats of all Intel floating point support conform to the proposed IEEE Floating Point Standard, insuring highly accurate results.

An important feature of the iAPX 88/10 processor is its ability to use a co-processor. The Intel 8087 is mounted on the iSBC 337 MULTIMODULE Numeric Data Processor to provide arithmetic and logical instruction extensions to the 8086 and 8088 CPU's. The instruction set consists of arithmetic, transcendental, logical, trigonometric, and exponential instructions which can all

operate on seven different data types. In many cases, the use of this MULTIMODULE board results in two orders of magnitude performance enhancement over a software solution. This board is the subject of a subsequent section of this application note.

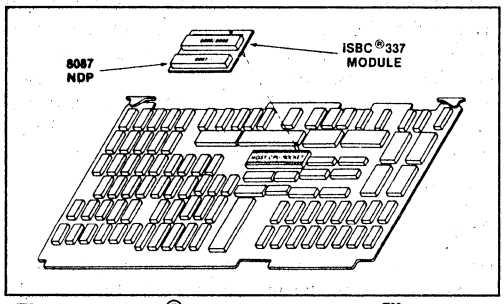

Figure 4. iSBC® 337 MULTIMODULE™ Numeric Data Processor

APPLICATION EXAMPLE

The features of the iSBC 88/40 Measurement and Control Computer can best be shown through an example. This application note describes the classical control system application of an agitated heating tank. Figure 5 shows the prominent features of this process control applications. The process consists of a storage vessel, a temperature sensor which measures the temperature of the fluid leaving the vessel, and a steam coil whose steam flow is regulated by a proportional valve. A motor drives an agitator to insure the temperature of the tank remains homogeneous.

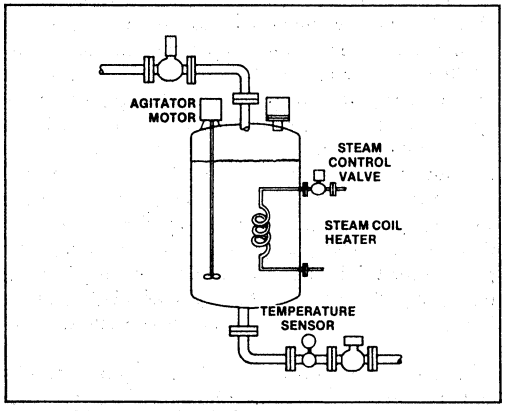

Figure 5. An Agitated Heating Tank

The passive portion of the application involves measuring the actual temperature of the fluid as it leaves the tank (and thus the temperature of all the fluid in the tank). If a control system is to be constructed which will control the temperature, an algorithm must be implemented which will provide control of the steam valve based upon the actual and the desired temperatures. This is the active portion of the application.

The control algorithm selected to control the tank temperature must be capable of compensating for disturbances created by a variety of conditions. For example, the temperature can be affected by changes in steam temperature, input temperature of the fluid, output flow rate, ambient temperature, and the flow rate of steam through the steam coil. Our control system will have control of only one of the variables and will only monitor the output temperature. To gain a degree of stability under these conditions, a feedback control algorithm is required. Alternatively, a system could be implemented using a feed-forward control algorithm. Unfortunately, the latter technique would require extensive instrumentation of all possible variables which could cause a disturbance. A feedback control system can take corrective action regardless of the source of a disturbance. Its chief drawback is that no corrective action is taken until an error is actually detected and, if not "tuned" correctly, some oscillations can occur.

Classical Controller Approaches

Before proceeding with a discussion of how a control system can be implemented using single board computers, a short discussion of classical control system theory is in order. This material will provide a background into the control algorithms which will be used as a basis for the digital control solution which will be developed.

The classical controller for feedback systems uses the "three mode" or PID (Proportional, Integral, Derivative) algorithm. In this system, the control output signal is a function of the error (the difference between the setpoint and the measured system variable). A specific application will use some combination of one, two, or all three terms making up the control statement.

Before continuing with the implementation of the control algorithm on the iSBC 88/40 Measurement and Control Computer, the various terms of the equation will be reviewed.

For Proportional control, the controller output is given by the equation:

$$m(t) = b + k_0 e(t) \qquad \textit{(eq. 1)}$$

where *m(t)* is the output signal, *b* is an adjustable bias value, k_0 is a gain constant, and *e(t)* is the measured error signal. Proportional control systems are normally

not used by themselves since corrections can not be made until an appreciable error has been detected. In addition, they tend to introduce oscillations into the system if the gain is set too large. Another disadvantage of proportional only systems is their inability to maintain a control element at some point (other than at its zero point using the bias term) in the absence of an error signal.

The second term in the PID solution is the Integral. The result of this term is to eliminate steady-state error or offset. The elimination of the offset is an important control objective; thus, the integral control term is widely used in conjunction with the proportional control element. The equation for the integral term is:

$$m(t) = (1/k_1)\, e(t)dt \qquad \textit{(eq. 2)}$$

where k_1 is the integral or reset time.

The Derivative term in the algorithm is used to provide an output which is a function of the rate of change in the error signal. It anticipates the future behavior of the system and improves the dynamic response to the controlled variable by decreasing the process response time. The format for the derivative term is:

$$m(t) = k_2(de/dt) \qquad \textit{(eq. 3)}$$

where k_2 is a constant representing the derivative time expressed in seconds or minutes. Because the output of the term is zero for a constant error, derivative control is never used alone in a control system. Instead, it is always used in conjunction with proportional and integral control. The derivative term is seldom used in flow controllers because derivative control tends to amplify "noise" which is picked up in the flow measurement, leading to an unstable control system. In addition, systems which have very large time delays do not benefit from the use of this term.

Implementation Using Digital Techniques

With an exposure to the fundamental concepts of control theory complete, the development of a solution using the iSBC 88/40 Measurement and Control Computer can proceed. A modular "top-down" approach will be used in this application note. The general requirements will be defined and "black boxes" will be developed to meet these requirements. Finally, the individual pieces will be combined to form a complete solution to the agitated tank control problem.

An effective control algorithm must deal not only with the mathematical solution of the control equation, but must also provide tests on limits and error conditions. As this application note will show, the iSBC 88/40 Measurement and Control Computer is easily able to support these additional requirements.

Additional supporting functions are also needed to effectively implement a complete control system solution. For example, provisions must be made to support input and update of the controller setpoints. Allowances must be made to modify control algorithm constants in order to "fine tune" the system after start-up. Raw analog data must be filtered to eliminate spurious sensor measurements and then must be converted into engineering units. In earlier system implementations not based on digital computers, these functions were performed using a "black box" approach. Here, each function is considered separately and the final solution is composed of combinations of building blocks.

Digital technology offers a simple analogy to this approach. Because application design is performed with software, a "black box" design is available for use with microcomputers. The black box corresponds to a software "task" and the system is integrated into a functional unit using a real time operating system. The iRMX 88 Real Time Executive provides all the tools needed by the software designer to implement his required functions for the application. This application note will show how the iRMX 88 executive can be used to simplify the design and to provide significant features in a process design example.

Figure 6 shows a block diagram of the operations needed to implement the control of one loop for the agitated heating tank. An attribute of using digital microcomputers is that additional loops can be run using the same hardware and software until the I/O or processing capabilities have been exceeded.

Each element of the block diagram represents one function which must be performed by the system. A task will be written to perform the functions assigned to each block. When the tasks are configured together with the iRMX 88 executive, a complete control solution will result. Some key features of the iSBC 88/40 Measurement and Control Computer will now be examined and a typical implementation will be described.

ANALOG SUPPORT FUNCTIONS

The information presented in Figure 6 indicates that many functions involve the manipulation of analog data and its conversion into a digital form usable by the processor. This involves the use of both hardware and software. This section of the application note demonstrates how the iSBC 88/40 board features can be applied to the solution of the analog portions of the system implementation. Both software programming concepts and hardware support products are examined.

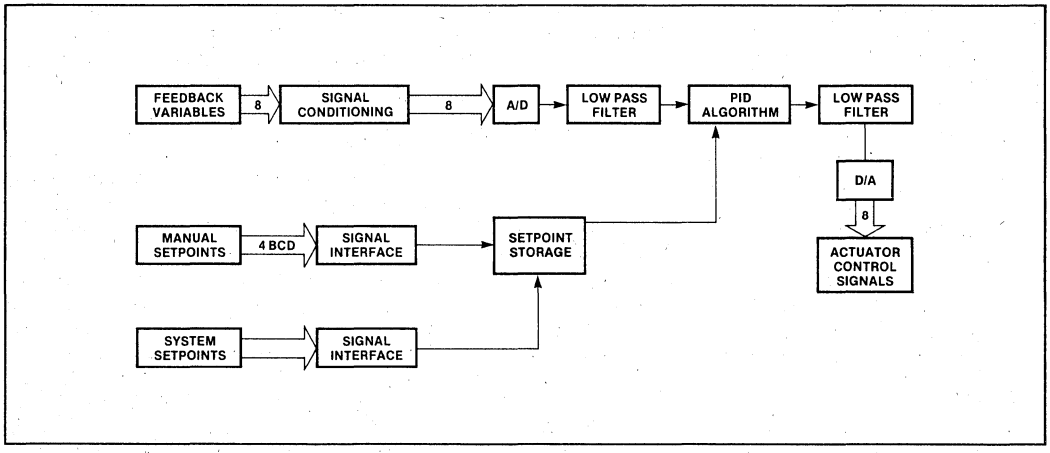

Figure 6. Control System Block Diagram

A digital computer performs most of the control system operations using software. Data is sampled from the process sensor and converted to an equivalent digital format. Subsequent operations use the digital form of the data. Unfortunately, this requirement for operating on sampled data, rather than continuous actual data, can lead to errors if the system is not properly implemented. Care must be taken to minimize errors when the original signal is digitized. Figure 7 shows how the digital signal may look when an analog signal is sampled using an analog to digital converter. A glance at the figure indicates that the error can be minimized by taking samples at shorter time intervals so that the staircase more closely resembles the original signal. Indeed, this is true, but what sample rate is best for a particular input signal?

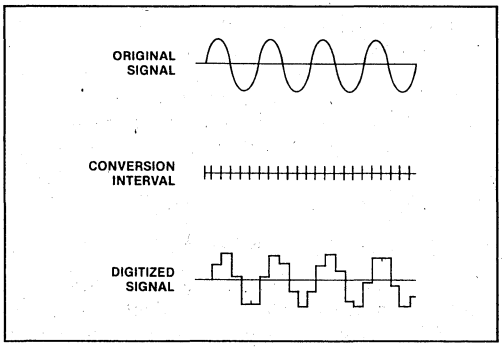

Figure 7. Analog Signal Digitization

A rule for digital control system designers is that the sample must be performed more than twice each period of the original analog signal. Thus, the sampling period must be less than one half the period of the sinusoidal frequency component which must be digitized. Even this method does not, in itself, assure an accurate measurement. Figure 8 shows the effects of the aliasing phenomenon on a high frequency signal. Aliasing converts the high frequency components into fictitious low frequency signals in the sampled results. Before data obtained from a digital system can be used, the unwanted signals must be filtered from the original sensor signal.

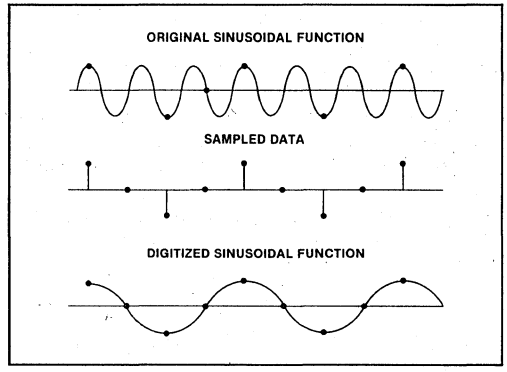

Figure 8. High Frequency Aliasing Error

Two approaches can be considered for filtering the data. One is the creation of an analog low pass filter and the second is the implementation of a digital filter. Unfortunately, a digital filter cannot remove aliasing error and is normally used to provide filtering of very low frequency oscillations. Analog filtering provides effective removal of unwanted frequencies but is expensive when attempting to gain sharp cut-off frequencies. A combination of the two technologies results in an ideal situation when used with digital controllers such as the iSBC 88/40 Measurement and Control Computer.

The final choice of sampling rate is usually determined by examining the process to be controlled. If a mathematical first order transfer function can be obtained for the process, either theoretically or experimentally, then the choice should be to use one tenth of the process time constant. If no function can be obtained and the frequency of the input signal is known and bounded, a sample rate equal to at least twice the input frequency is used. If none of the above is known, a rough estimate for process applications is to use a 1 second sample period for flow measurements, a 5 second interval for level or pressure measurements, and a 20 second interval for temperature or composition measurements. In any case, faster sampling than is necessary is a waste of computing power and limits the number of PID loops that can be supported by a given system.

The elimination of high frequency noise in systems using Intel's control products is best accomplished using the iCS 910 Analog Termination Strip. This strip has provision for the installation of a single pole RC low pass filter (details on the use of this strip in industrial control applications can be found in AP-52, Using Intel's Control Series In Industrial Applications). In addition to providing a front-end low pass filter, the strip gives a simple method of terminating analog wiring to the analog to digital converter. Figure 10 indicates the cable connections which can be used to connect the analog input connectors of the iSBC 88/40 board to the iCS 910 termination strip. This connection arrangement will provide complete compatibility between the numbered channels on the termination strip and those defined by the measurement and control computer.

The iSBC 88/40 board's application software can be used to eliminate the effects of low frequency noise in the sampled signal. This is done by implementing a simple digital low pass filter. The equation for a first order filter is:

$$Sf = a(Sm) + (1 - a)\ (Sf') \qquad \textit{(eq. 4)}$$

where *Sf* represents the filtered output, *a* is a function of the cutoff frequency, *Sm* is the measured sample, and *Sf'* is the last filtered output result. If additional poles are required, the equations can be cascaded as required.

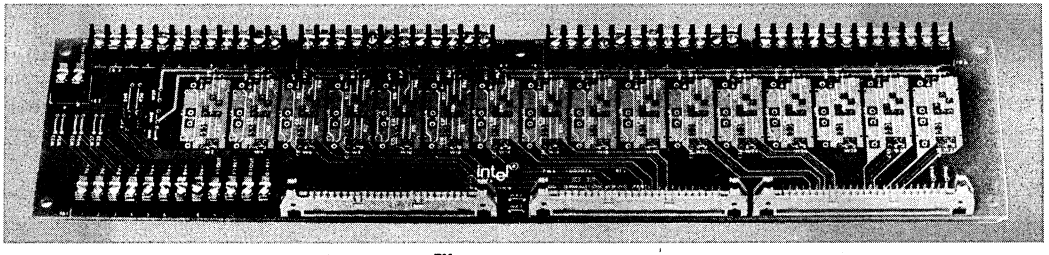

Figure 9. iCS™ 910 Analog Termination Strip

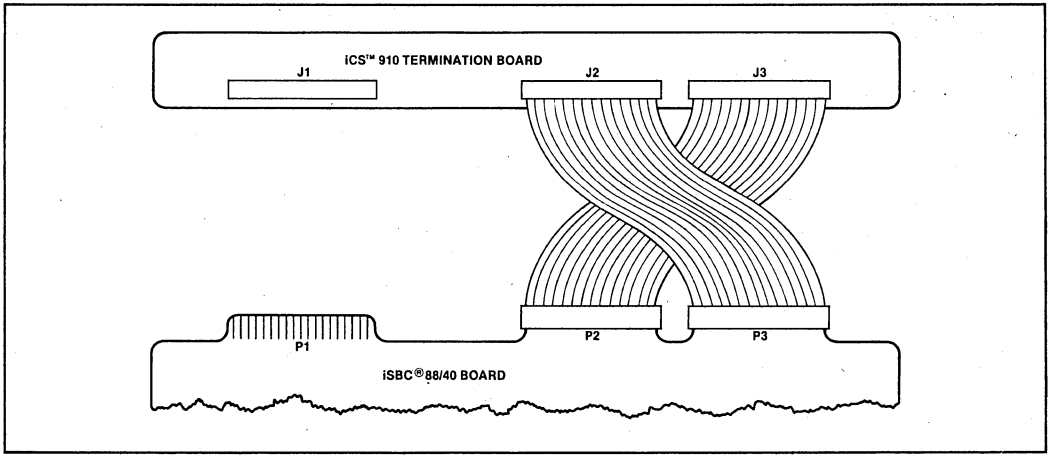

Figure 10. Termination Board Interconnects

The implementation of this filter using Intel's PL/M-86 high level language is straightforward. A simple procedure can be written in which the measured value, the last filter output value, and the value for *a* are passed with the call. The procedure returns the new filtered value. The code for such a procedure is shown in Figure 11. Note that the computation is performed in steps to prevent any stack overflows from occurring when real numbers are used. This should be done whenever the algebraic equation exceeds eight terms. The 8087 stack used in internal operations can overflow when more than eight operations are nested together. Breaking the equation into smaller steps can prevent any overflow errors from occurring.

```
1       Analog$filter$module: Do;
2   1   Analog$filter:
          Procedure (present$value, last$output, cutoff) real public;
3   2     Declare (present$value, last$output, cutoff) pointer;

4   2     Declare New$signal based present$value real;
5   2     Declare Old$filter based last$output real;
6   2     Declare Alpha based cutoff real;

7   2     Declare New$filter real;

8   2     Declare temp1 real;
9   2     Declare temp2 real;
10  2     Declare temp3 real;
11  2     Declare One real data (1.0);

12  2     temp1 = Alpha * New$signal;
13  2     temp2 = One - Alpha;
14  2     temp3 = temp2 * Old$filter;
15  2     New$filter = temp1 + temp3;
16  2     Return New$filter;

17  2   end Analog$filter;
18  2   end Analog$filter$module;
```

Figure 11. Low Pass Filter Algorithm

Before data can be sent to the filter, it must be converted into floating point format and then into engineering units. The conversion into engineering units can involve a complex algorithm if the raw data is non-linear. The design of future systems can be simplified if the programmer generates procedures which are general enough to cover the majority of cases found in his application environment. The following example shows how the iSBC 88/40 board can be programmed to provide the linearization and conversion for the general case.

LINEARIZATION FUNCTIONS

The program developed for this application example uses an interpolation technique. A table look-up enables a program to be written which will support both linear and non-linear analog sensors. The number of entries in the table is a function of the desired resolution and of the non-linearity. For example, linear functions needing only scaling and offset ($y = ax + b$) require only two table entries. A separate table is maintained for each sensor channel. The program is written to support a maximum of 256 entries per channel which should provide at least 0.1 percent accuracy for all but the most non-linear applications.

Each table entry consists of a raw value and a corresponding real engineering unit value expressed in floating point format. The linearization program's declaration of such a table is shown in Figure 12. The application software must determine the bracket or location of the terms in the table which lie above and below the raw input value. The algorithm to find the bracket in the table which corresponds to the raw data input can be programmed as shown in Figure 13. Once the bracket has been found, the actual engineering value can be calculated and passed back to the calling program. The code for performing the interpolation calculation might look like that shown in Figure 14. Data for the tables can be determined from known characteristics of the sensor or a program can be written which allows the user to enter known points into the table dynamically during calibration. In this application note, an assumption is made that the data has been entered into the table from known characteristics rather than actual calibration.

```
6   2   Declare (table based table$pointer)(255) structure (
            x word,
            y real );
```

Figure 12. Declaration of Table

```
10  2   Do while table (n).x < raw$value;
11  3     n=n+1;
          /* special case, above table */
12  3     If n > table$entries
          then do;
14  4       eu=table(n-1).y;
15  4       return eu;
16  4     end;
17  3   end;

          /* special case, below table */
18  2   If n=0
          then do;
20  3     eu=table(n).y;
21  3     return eu;
22  3   end;
```

Figure 13. Bracketing Algorithm

```
                /* interpolate engineering units */
23    2         dx=float(int(table(n).x-table(n-1).x));
24    2         dy=table(n).y-table(n-1).y;
25    2         dr=float(int(raw$value-table(n-1).x));
26    2         eu=dr * dy;
27    2         eu=eu / dx;
28    2         eu=eu+table(n-1).y;
```

Figure 14. Interpolation Algorithm

One final component of the analog design which is required is the creation of software which will actually interface with the analog to digital converter and transform data from the analog world into a digital domain. Again, a program should be developed which is general enough to handle a wide variety of applications. It should be compatible with both the on-board A/D sections and with the iSBC 311 Analog Input MULTIMODULE Board, which may be installed for analog expansion.

The interface with the analog portions of the boards is easily handled using software. The ADC can be commanded to select the desired analog channel and begin a conversion by sending the appropriate byte containing the channel and gain bits to a port corresponding to the ADC. When using the on-board converter, the iSBC 88/40 board user should send the command byte to port 0D8 hex. The actual selection of the desired channel and the conversion takes only 50 microseconds, so little is gained by using an interrupt instead of status testing to detect the end of conversion. The status bit is tested by reading the input status port (0D8 hex for the on-board converter). When the conversion is complete, the bit will have a value of 0.

Certain multiplexer components used in the ADC require that a delay time be added to the basic 50 microseconds for the channel to settle after a new gain setting has been selected before reading the sample and hold converter. The amount of delay is a function of the gain and varies from 0 (gain = 1) to 30 milliseconds (gain = 250). The analog driver software must take this settle time into account. Figure 15 shows the required settle times for the various gain settings. The delay is easily implemented using the facilities of the iRMX 88 nucleus. While the system is waiting for the settling time, other tasks can use the processor to execute their code. Figure 16 provides an example of a program which gets data from the analog to digital converter for a selected channel and gain. The iRMX 88 request for a time delay is implemented using the call to RQWAIT specifying the desired delay. In the example, the system delay increment is assumed to be 5 milliseconds, so the required number of delay increments is specified as 6 in order to wait for 30 milliseconds at high gains. Note that, for gains of one, the delay is skipped. After the required delay has elapsed, the converter is again activated using another output to its command port. This output must again include the channel and gain information.

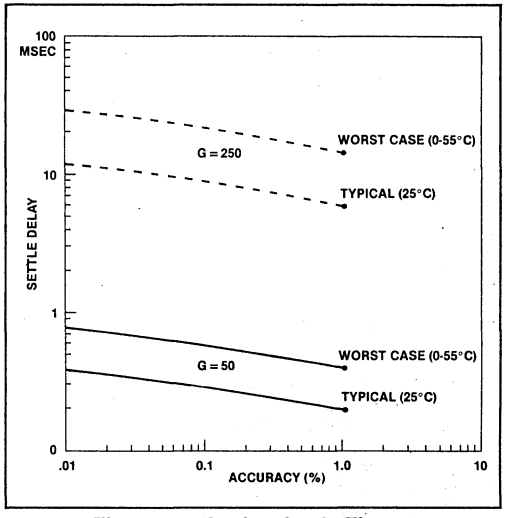

Figure 15. Analog Settle Times

```
                /* select mux channel */
14    2         output(port$adr)=channel or gain;
15    2         if gain < 40h
                then do;
                  /* settling delay for high gains */
17    3           msg$ptr=rqwait(.timeout, 6);
18    3           output(port$adr)=channel or gain;
19    3         end;
                /* wait for end of conversion */
20    2         do while (input(port$adr) and 01h) > 0; end;

                /* get adc data */
22    2         low$raw$data=input(port$adr) and 0f0h;
23    2         high$raw$data=input(port$adr+1);
24    2         raw$data=shl(high$raw$data, 8) or low$raw$data;
```

Figure 16. Analog Input Routine

A workable analog driver must provide more than just the ability to get data from a specified channel. At a minimum, the zero offset induced by the temperature of the circuitry must be removed from the raw data. In some cases, an additional correction is required to compensate for gain error induced by temperature. However, the effect of the latter is small and can usually be ignored.

Provisions are included on the iSBC 88/40 Measurement and Control Computer to simplify the task of providing a zero offset correction. Wire-wrap stakes are mounted

on the board to facilitate grounding one of the input channels. In the differential mode of operation, channel 15 represents the zero reference offset voltage. If a data channel has the offset subtracted from it, the result will be a value which is compensated for offset drift and which is highly accurate over a wide range of board temperatures. Figure 17 shows the software which can be used to collect data from a channel and which will deliver a zero compensated value to a calling program. In Figure 17, note that the values are converted to an offset binary representation to be compatible with the standard output of the analog to digital circuitry.

```
     2   zero$data = get$channel (gain, ref$chan, port$adr);

         /* get data channel */
35   2   raw$data = get$channel (gain, channel, port$adr);

         /* support negative offset */
36   2   if zero$data > raw$data
         then do;
38   3     raw$data = zero$data - raw$data;
39   3     raw$data = 8000h - raw$data;
40   3   end;
41   2   else do;
42   3     raw$data = raw$data - zero$data;
43   3     raw$data = raw$data + 8000h;
44   3   end;
```

Figure 17. Zero Compensation Procedure

The analog input driver required for the application can now be constructed using the software building blocks which have been created. Generally, the input data will consist of either thermocouple inputs or non-temperature sensitive inputs. The driver must be able to support both by providing a selective cold junction compensation correction for those channels which are designated as thermocouples.

The problem is illustrated in Figure 18. The voltage which represents the temperature of the thermocouple consists of the sum of the actual thermocouple voltage plus the voltage which is generated by the thermocouple junctions created where the wiring is terminated. The error introduced by the termination must be removed before a junction temperature can be calculated. If the thermo/voltage characteristics of the termination junction are known, the induced error can be subtracted and the temperature of the thermocouple can be calculated.

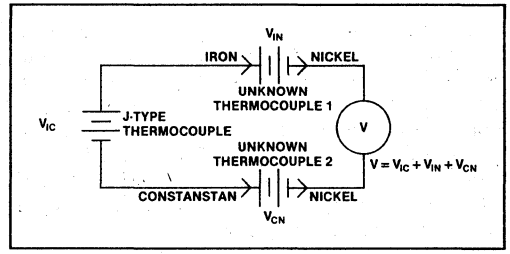

Figure 18. Thermocouple Cold Junction Error

Two things must be known for the correction voltage to become available. First, the actual temperature of the junction board must be known. Second, the electrical characteristics of the junction with respect to temperature must be defined. With this data available, the correction voltage can be obtained using the linearization program which has been created as an analog building block.

The first problem is solved by installing a temperature sensing circuit onto the iCS 910 Analog Termination Strip. Figure 19 shows such a circuit which can be used to provide an extremely accurate measurement of the board and terminator temperature. Note that the circuit is installed onto the termination board using the mounting locations originally designed for the installation of a low pass filter. The output of the temperature sensing is

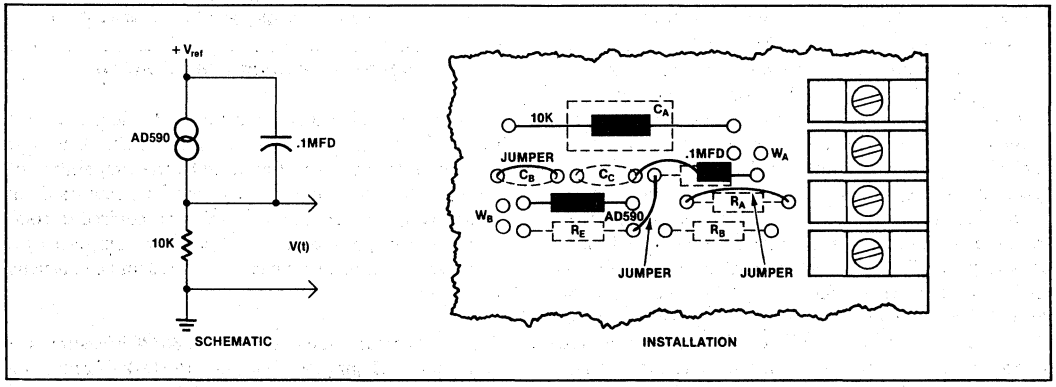

Figure 19. iCS™ 910 Board Sensing Circuit

related only to the temperature of the sensor device which provides a current of 1 microamp per degree Kelvin through the 10K resistor. The temperature is related to the voltage by the equation:

$$V = (273 + T) / 100 \qquad (eq.\ 5)$$

Thus, the voltage read from the termination strip as the temperature varies from 0 to 70 degrees Centigrade will vary from 2.73 volts at 0 degrees to 3.43 volts at 70 degrees. The analog to digital converter should operate at a gain of one to read this voltage. This will provide a resolution (1 bit change) of 0.70 volts / 0.00244141 volts/bit or 286 bits/70 degree change. This equates to about 0.25 degree per bit change.

The second problem is solved by connecting a thermocouple, which is placed in an ice bath, to the iCS 910 strip. The strip is placed into an environmental chamber and the output monitored as the board and junction temperature is varied. The output represents the correction required at each temperature. Tests made for this application note indicated that the error was essentially linear over the board range from 0 to 70 degrees Centigrade. The correction voltage was found to vary linearly from minus 0.102 millivolt at 0 degrees to 3.578 millivolts at 70 degrees. This data was placed into a linearization table to give an offset correction for a measured temperature of the board. Figure 20 shows the table and the code required to correct the raw temperature value from thermocouple inputs.

```
                /* get thermocouple reference junction temp */
60    7         j$raw = analog$to$digital$conversion (
                  gain$one,
                  13,
                  channel$data, port$number );

61    7         tc = analog$linearization (
                  @cold$junction$table,
                  2,
                  j$raw );

62    7         tc = analog$filter (
                  @tc,
                  @channel$data, last$thermocouple,
                  @channel$data, filter$cutoff );
63    7         raw = raw + unsign(fix(tc));
      7         channel$data, last$thermocouple = tc;
```

Figure 20. Thermocouple Correction Program

An analog input driver can now be constructed which is compatible with a variety of applications. It will run as a task under the iRMX 88 nucleus. In order to support up to "n" analog inputs, an exchange is used to store information about current active analog channels. User tasks requiring analog facilities send a request to the analog exchange indicating the parameters of the desired channel. Because an exchange has a FIFO storage capacity for messages, each active channel is sampled by the task in turn, then placed back onto the exchange. A unique message is used to indicate the beginning of the channel requests. Figure 21 provides a partial listing of the code used to make up the analog input task.

```
57    3   msg$ptr = rqwait (.timeout, 2);

58    3   last$channel = false;

59    3   do while last$channel = false;

60    4     msg$ptr = rqwait (.analog$exch, 0);
61    4     if channel$data, type = null$type
            then last$channel = true;
63    4     else do;

              /* test for conversion time request */
64    5       if channel$data, conversion$counter = 0
              then do;

                /* get raw data from adc */
74    6         raw = analog$to$digital$conversion (
                    gain,
                    channel$number,
                    port$number );

                /* perform engineering unit conversion */
83    6         eu = analog$linearization (
                    table$pointer,
                    number$of$entries,
                    raw );

                /* filter the data */
84    6         eu = analog$filter (
                    @eu,
                    @channel$data, last$value,
                    @filter$cutoff );
85    6         channel$data, last$value = eu;

86    6         channel$data, conversion$counter
                = conversion$interval;

87    6         exch$ptr = channel$data, output$exchange$ptr;
88    6         data$ptr = rqwait (exch$ptr, 0);
89    6         data$message, value = eu;
90    6         call rqsend (exch$ptr, data$ptr);

91    6       end;

              /* decriment counter if not ready yet */
92    5       else channel$data, conversion$counter =
              channel$data, conversion$counter - 1;

93    5     end;
94    4     call rqsend(.analgo$exch, msg$ptr);
```

Figure 21. Analog Input Task

Updated data is stored in an output exchange in order to assure mutual exclusion of the engineering unit conversion of the data. Mutual exclusion guarantees that the data cannot be read by another task while it is being up-

dated (during the updating process, multiple bytes of data must be changed; until all are modified, the number cannot be considered valid). The exchange mechanism of iRMX executives supports the movement of messages (this might be compared to a letter in a mailbox). If the data is stored as a message in an exchange, it is available to the first user requesting it. While that user has the message (letter), it is not available to anyone else. When he is finished with it, he will return it to the exchange so that other users may operate upon the data. Note that the sample interval of each channel is selected by the requesting task so that optimum processor efficiency can be obtained.

Certain parameters used by the analog input task must be retained even if the system power is shut off for an extended period of time. These parameters are used to provide the task with unique information such as the channel and port address, the desired gain, the conversion interval and the linearization and engineering conversion data. On the other hand, some information used by the task can be easily created dynamically and does not require the use of non-volatile storage. Examples of the latter category include addresses of the storage exchanges and addresses of the various messages.

The use of E^{2}PROM on the iSBC 88/40 Measurement and Control Computer provides the mechanism for the storage of those parameters which must be occasionally modified. Figure 22 shows a possible technique for passing the analog input task its required information and pointers to the non-volatile data. Intel's PL/M-86 provides a convenient mechanism for referencing variables whose physical location is passed as a parameter. This is the BASED VARIABLE. A declaration is made which indicates the location of the variable containing the address of the data. For example:

```
Declare CONSTANT$POINTER pointer;
Declare CONSTANT based
  CONSTANT$POINTER real;
```

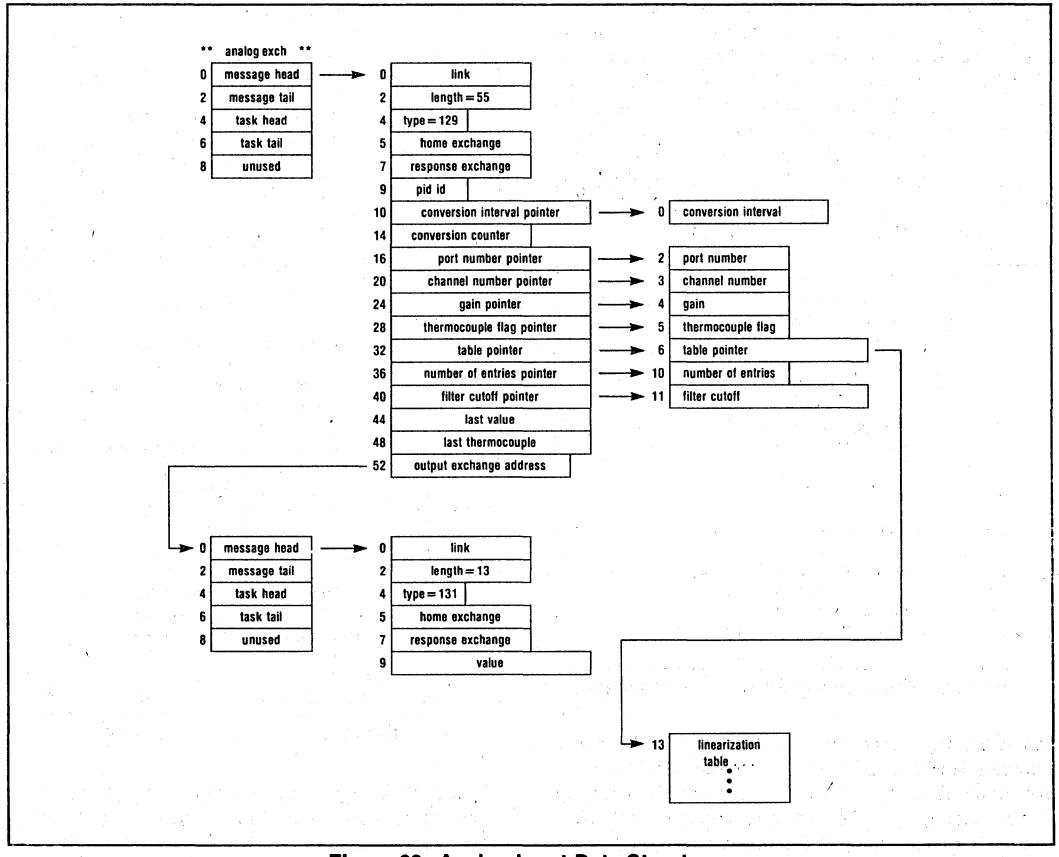

Figure 22. Analog Input Data Structures

The CONSTANT$POINTER contains the address of the constant which is to be used in the calculations. Any program reference to CONSTANT will cause the processor to use the real number stored in the address pointed to by CONSTANT$POINTER.

This technique allows a message to contain pointers to E^2PROM constants which can be used by the task in performing its functions. Indeed, multiple levels of based variables can be used as shown in Figure 22.

CONTROL ALGORITHM

The implementation of a control algorithm on the iSBC 88/40 computer involves more than just implementing the PID equation. To become truly cost effective, multiple loops must be supported by the board and error checking/correction must be included. As with the analog input functions, system control parameters must be maintained in non-volatile memory. Finally, the system must be capable of operating in real time with a minimum of required processor time. This section examines some of the features which are used to provide these functions on the measurement and control computer.

The first design goal is to support multiple control loops using as little of the processor's time as possible. The processing time is minimized by performing all complex mathematical computations using the 8087 math co-processor mounted on the iSBC 337 MULTIMODULE board. Certain details of the software implications of the co-processor are important to the system designer.

In many cases, the iRMX 88 nucleus will provide all the required initialization operations for the co-processor chip. The nucleus sends a default control word setting the device to mask all exceptions and interrupts, define a 64 bit precision, and to round up all operations. In an iRMX environment, the application programmer has no need to send additional mode commands to the processor. However, the mode can be changed by using the PL/M built-in procedure, SET$REAL$MODE, if required. In the application code written for this application note, certain conversion algorithms required that results obtained from the math operations be truncated. To instruct the 8087 to perform this truncation, a command word of 0FBF hex is sent in the initialization segment of a task.

Multiple control loops are implemented using the iRMX 88 exchange mechanism. Here, messages are queued at an exchange in a first in, first out (FIFO) manner. One message can be sent to the exchange for each control loop to be executed. A special message is placed into the exchange at control task initialization to be used as a pointer to the end of the queue. Each time the control task is to run, it will read messages sequentially from the exchange until it encounters its end of queue message. Each message corresponds to one control loop's specifications and is returned to the exchange when the loop has been completed. A separate control interface task manages the control loop activation by sending a message containing the necessary parameters to the control exchange. This technique allows the interface task to also remove a control task by taking the appropriate message from the exchange when parameter modifications or control loop deletion is requested.

Each message at the control exchange (in the application example, this exchange is called PID$EXCH) contains pointers to various other exchanges or data structures. The relationships of these structures is shown in Figure 23. Note that some system parameters should be stored in non-volatile E^2PROM memory. System constants are stored in an exchange pointed to by the primary control message. An exchange is used here so that the system can provide mutual exclusion of the data if it is required to modify one or more of the parameters while the control system is running. Additional exchanges are used to store the input and output terms in order to insure compatibility with the analog input and output tasks.

In order to function correctly, a digital implementation of a PID control algorithm requires operation at a known time interval. In the case of the implementation constructed for this application note, a time increment of 100 milliseconds was desired. The iRMX nucleus provides the ability to perform a timed wait at an exchange via the call to the primitive procedure, RQWAIT. Unfortunately, this procedure can not be directly used in the task to provide the required task delay. This is because the execution time of the task is a function of the number of loops being implemented and also varies slightly depending upon the program paths required by the data values. Thus, a mechanism must be implemented to provide the task synchronization.

The desired time delay can easily be obtained by using an associated synchronization task. In this task, the RQWAIT primitive can be used with the required time delay. Because the task execution time is not a variable, this task can be used to provide synchronization for its supported task. Figure 24 shows how the two tasks can communicate with each other. Two exchanges are maintained. One, called the PID bucket exchange in the implementation, is used by the main task to indicate that it is beginning its execution and that a new time period delay is to begin. The timer task (whose priority should be greater, i.e., having a smaller priority number) will wait at the bucket exchange for the message. When it is received, it will begin a delayed wait at an exchange. When the timeout period has elapsed, the message is sent to a second exchange (in the figure, this exchange is

called the PID trigger exchange). The main task, after completing the servicing of all operational PID control loops, will wait at the trigger exchange for a message from the timer task. In the case of the application example, the message will arrive 100 milliseconds after the task began its last update of the control loops.

When iRMX 88 timed wait operations are implemented on the iSBC 88/40 Measurement and Control Computer, timer 0 of the on-board 8253 programmable interval timer must be used. A wire wrap jumper must be installed to vector the output of the timer to one of the interrupts of the 8259A programmable interrupt controller chip.

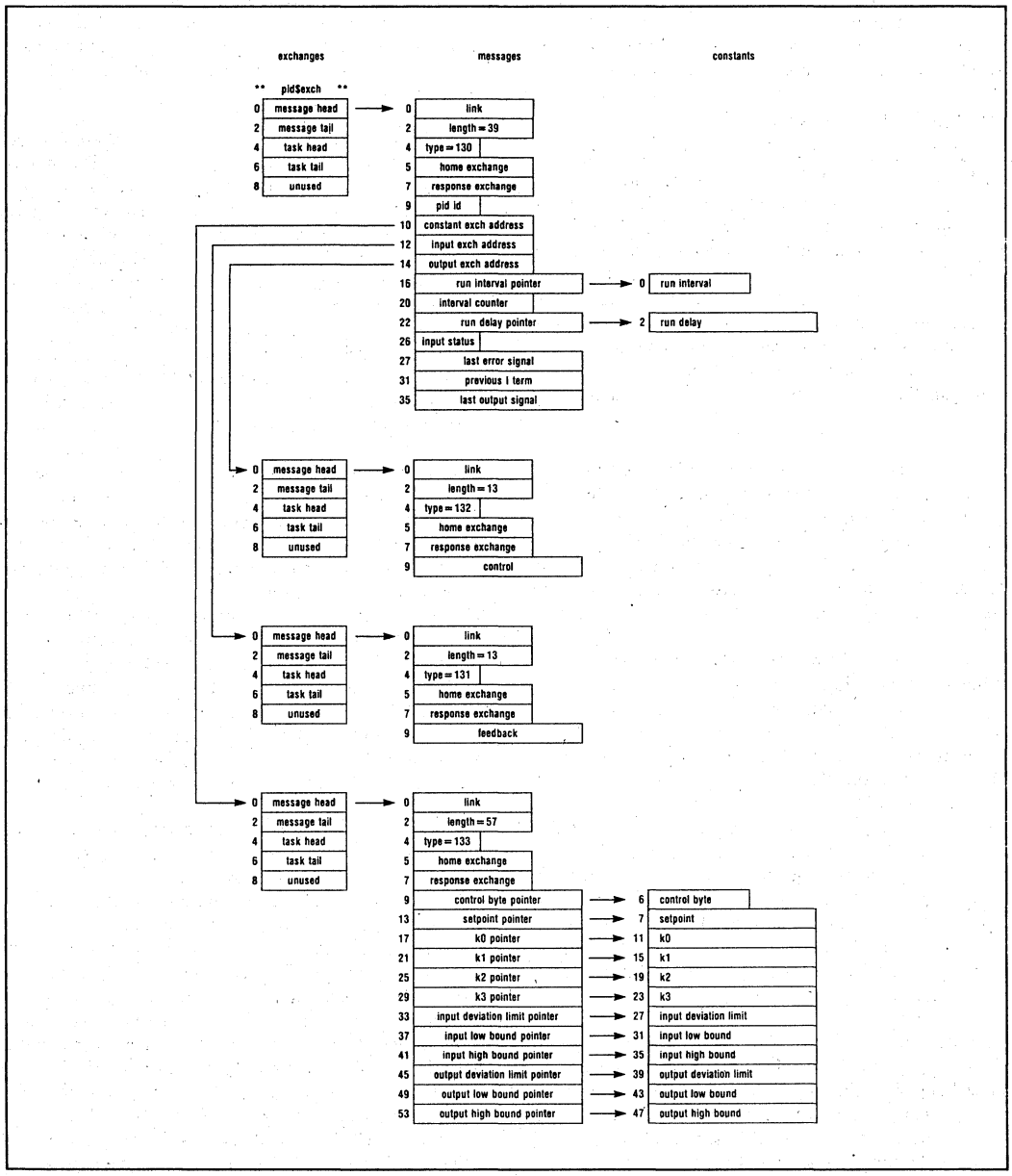

Figure 23. Control Structure Relationships

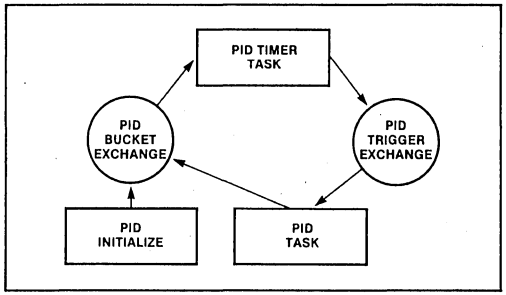

Figure 24. Synchronization Task

Normally, interrupt level 1 is used for this timer; however, any available level may be selected and the iRMX nucleus can be configured to operate correctly by the Interactive Configuration Utility (ICU). A later section of this application note will explain the ICU interaction in more detail. The timed delay function will allow delay increments as small as one millisecond. Each call to the delayed wait function specifies the number of delay increments for which to wait (0 to 65535 increments may be specified in the call). Care should be taken when specifying a delay period of only one unit. The system is not always capable of resolving this delay accurately and an indeterminant delay of from 0 to 1 unit may actually elapse.

ANALOG OUTPUT FUNCTIONS

Once the feedback loop of a control system has been sampled and a control signal generated by the PID control algorithm, a final conversion must be made to provide an output compatible with the system control element. Because such a wide variety of control elements is available, the iSBC 88/40 board was designed to accept the output control circuitry as an expansion option rather than to build unnecessary components and drivers into the board.

In most cases, the control element is driven with an analog signal, either a 4-20 milliamp current signal or a DC voltage. The iSBC 88/40 board is easily interfaced to the analog world using the iSBX 328 Analog Output MULTIMODULE Board. The use of this board is so common with the measurement and control computer that it warrants the time to explain its operation in some detail.

Each iSBX 328 board provides up to eight voltage or current outputs which can drive a wide variety of control devices.

The use of intelligence on the MULTIMODULE expansion board is a key element providing the ability to incorporate eight channels in a very small physical area. The capabilities of the intelligence allow the board to provide significant enhancements to the basic operational characteristics of the host iSBC 88/40 board. One example is the ability to perform diagnostics of the analog output module upon command from the 8088 processor on the host measurement and control board.

The expanded capabilities bring with them a requirement for some care on the part of the system designer. A fixed programming sequence and handshaking are required to reliably communicate with the expansion board. An analog output driver is easily written which provides the necessary support for analog output signals associated with the control application.

Each time a reset is issued to the iSBX 328 board, it executes a test of its internal stored program to assure data integrity and of its usable RAM to insure that each location can be written to and read from correctly. If either of these tests fail, a bit in the status field will be set to indicate the failure. If the test is performed satisfactorily, the F0 status register (bit 2 of the base port + 2) is set to indicate that the board is ready to receive an initialization command. The initialization command is used to specify the operational mode and number of channels being used.

The operational mode specifies which of four internal programs are to be used to move data between the iSBX interface and the outside world. Each program specifies a unique hardware configuration of the board. Two programs are associated with unipolar operation of the DAC outputs. Program 1 is used when some channels are associated with voltage outputs and some are configured as current outputs. Data directed to a current output will be internally scaled and offset before being sent to the DAC. Data specified as directed to a voltage output is not modified by the program. Program 2 indicates that either all eight channels are used for voltage outputs or that all eight channels are used for current output. Current outputs will be offset by the hardware but no scaling is accomplished. This program 2 mode results in a 10% increase in performance over what is specified in the data sheet of the iSBX 328 board.

Both unipolar programs assume that the data is pure binary formatted with a 0 hex corresponding to a voltage level of 0 volts (or 4 milliamps). A value of 0FFF0 hex will generate a voltage of 4.99 volts (in the current configuration mode, program 1 will result in a 20 milliamp current while program 2 will result in an output of 24 milliamps).

There are also two operational modes which can be used to support a bipolar operation. Program 2 provides a direct hardware support capability for those cases where

all outputs are either configured as entirely voltage or entirely current outputs. No adjustments are made to the data prior to being sent to the DAC. The data format used for both bipolar modes is the offset binary representation of a number. Negative numbers are represented by the values 0 (−32752) to 8000 hex (0). Positive numbers range from 8000 hex (0) to 0FFF0 hex (+32752). Channels defined as current outputs have no legal negative output values.

Finally, program 4 is used to support bipolar operations where the outputs are mixed between current and voltage. The program does not alter data destined to voltage channels but does offset and scale data for channels designated as current outputs.

Once the device has been initialized, subsequent data transfers are all through the data transfer port located at the base address of the board's MULTIMODULE socket. Before each write to the device, the driver software must check the IBF bit (bit 1) of the status to verify that the input buffer is not full. An additional bit is used to specify to the host processor which data byte (high or low) is next to be passed. The low order bits of the low data byte specify which channel the data is for and also what configuration (voltage or current) corresponds to that channel.

Like the analog input task, the application driver for the analog output can be an iRMX 88 task using exchanges and messages for data transfer. In the example implemented for this application note, an exchange, DAC$EXCH, was dedicated to the control of the task. It contains messages which specify the output port, channel used, and output mode (current or voltage). The task runs at a twenty millisecond time interval and updates each channel as indicated by the control messages. The location of the exchange used to store the output data is also specified by the control message. The use of this exchange mechanism provides mutual exclusion of the output data.

The external connections to the iSBX 328 analog output board can be made using the Intel iCS 910 Analog Termination Strip. When used in this mode, the analog outputs are available on the terminal strips originally designated as analog input channels. Figure 25 shows how the interconnect cable can be used to install the termination panel to the board.

E²PROM FUNCTIONS

Several references have been made to the advantages gained by using E²PROM 2816 devices on the iSBC 88/40 board for the storage of non-volatile variables. Many configurations mixing E²PROM devices with combinations of EPROM, ROM, and/or byte wide RAM are possible. Support is provided for the installation of one, two, four, or eight (using the iSBC 341 MULTIMODULE EPROM board) 2816 devices. Complete E²PROM write capability is provided on the board. The Intel supplied hardware for this support includes a switching power supply and wave-shaping circuitry. Only minimal user programming overhead is required by the application program.

The on-board wave-shaping circuitry provides a 2816 compatible programming pulse of approximately 16 milliseconds duration. In order to generate this pulse, use is made of the on-board 8253 programmable interval timer. Wirewrap jumper posts are provided to route the timer output to the pulse generator. The gate to the timer is connected using an additional jumper to the memory decode logic to signal a write request to a 2816 device. User software must be provided to program the 8253 for the generation of a 14 millisecond pulse. This code is most easily located in the initialization portion of one of the application tasks associated with writing data into

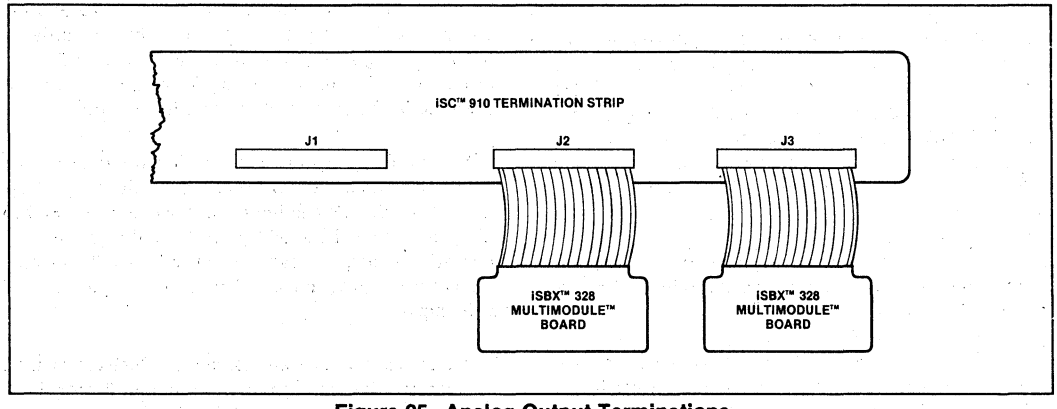

Figure 25. Analog Output Terminations

the devices. Only three lines of PL/M-86 code are required to perform the initialization. The application example includes the code:

```
output(0D6H)=0B2H;
  /* timer 2 to mode 2 */
output(0D4H)=000H;
  /* most significant byte */
output(0D4H)=038H;
  /* least significant byte */
```

The hardware will now generate the appropriate programming pulses to write into the 2816 each time data is written into an address occupied by the device. When the EPROM size is larger than 2K bytes size of the 2816, the system will create a duplicate image of the 2K block as many times as is required to fill the size specified for the EPROM. For example, if 2732A EPROM devices are used and one 2816 is installed at a base location of 0F8000 hex, one image of the E²PROM data will occupy the memory from 0F8000 hex to 0F87FF hex while a second image will be seen from 0F8800 hex to 0F8FFF hex. Reads or writes to either image will access the same data and either may be used.

The user must consider the possibility of system power failures and their impact when designing systems which use the iSBC 88/40 board's E²PROM capabilities. This is especially true in systems whose power supply for the +5 volt source is protected by a crowbar circuit. The onboard switching power supply which generates the high voltage programming pulse operates at very low input voltages and its RC time constant will provide significant voltage levels even if the +5 volt input supply is abruptly removed. The presence of a programming voltage in the absence of a +5 volt supply to a 2816 can cause irreversible damage to the E²PROM chip. The potential for this condition during a write cycle must be considered by the designer. Figure 26 shows a circuit which can be added to the system and connected via the iSBC 88/40 board's P2 connector if desired. The purpose of the circuit is to crowbar the V_{pp} programming voltage to the +5 volt supply if the +5 volt voltage level drops below about 4.5 volts, thus preventing any damage to the 2816.

From a software standpoint, only two items need be given attention during the writing of E²PROM devices on the board. First, before any location can be written in the 2816, the location must first be cleared to an initial value of 0FF hex. Unless this value is already present in the device, two write cycles are required to store new data (the 2816 has a chip erase mode but it is not supported on the iSBC 88/40 board). The second item involves inhibiting interrupts during the write cycle. The programming pulse generation circuitry uses the onboard timeout circuitry, so the timeout interrupt, if used, must be masked off prior to beginning the write cycle (this implies that the hardware for the timeout acknowledge must be installed to all the circuitry to become a part of the pulse generator). In an iRMX 88 environment using timed waits, the interval timer must also be masked off during the write cycle. If these interrupts are not masked off, the processor can respond to an interrupt and begin modifying its internal registers which point to the memory. This will result in incorrect programming of the E²PROM device. An example of the code which might be used to program a 2816 is shown in Figure 27. The programmer should keep in mind that, during the programming of the 2816, the iAPX 88/10 processor is in a wait state and cannot process any instructions. Thus, for each byte written, approximately 36 milliseconds must elapse before processing can again begin (18 milliseconds for the clearing of the byte and another 18 for the data write). If timed waits are being performed, an error will be introduced into the system. Some critical applications may need to take this into account.

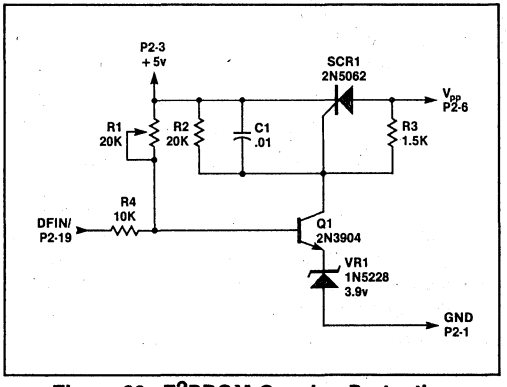

Figure 26. E²PROM Crowbar Protection

```
                then do;
366   5           disable;
367   5           call movb(@erase$pattern(0),
                  @constants(k).table$pointer,4);
368   5           constants(k).table$pointer=
                  @constants(k).linearization$table(0);
369   5           enable;
370   5         end;
```

Figure 27. E²PROM Programming Example

System Implementation

The application programs for the example described in this application note have been implemented using Ver-

sion 1.1 of the iRMX 88 executive. The use of the Interactive Configuration Utility (ICU88) considerably reduces the effort required to bring a system on line by providing a question and answer session with the programmer. The output of the ICU consists of a system configuration module and a submit file which provides most of the required LINK and LOCATE commands.

In the application, a system time wait increment of 5 milliseconds was chosen. Figure 28 shows the dialogue required to implement the timed wait feature using the board with the output of channel 0 (from the 8253 programmable interval timer) connected to interrupt level 2. Note that the system time unit of 6140 corresponds to a 5 millisecond increment.

```
INTERRUPTS: Y
  8259A PORT: C0H
  8259 INTERVAL: 2
  INTERRUPT SERVICE ROUTINE VECTOR BASE: 56
  TIMED WAITS: YES
    TIMER LEVEL: 2
    8253 PORT: D0H
    SYSTEM TIME UNIT: 6140
```

Figure 28. ICU Timed Wait Dialogue

Additional entries into the ICU define the system tasks and their associated exchanges. The desired locations of the RAM and EPROM are specified and the configuration modules created. The location of the E2PROM module is specified in one of the user created modules as a public pointer which is initialized with the base address of the device. An example might be:

```
e2prom$module: do;
  declare e2prom$pointer pointer public
    data (0f8000h);
end e2prom$module;
```

All references to the data structures in the 2816 are by means of a based variable or structure.

Before executing the submit file for a ROM based system, it is necessary to edit the LOCATE command to include the BOOTSTRAP request. This will assure that the locator places a long jump at the reset vector location when the system is executed out of EPROM. Execution of the LOCATE facility will generate a warning 38 which should be ignored. The corrected LOCATE code is shown in Figure 29.

```
ISIS-II MCS-86 LOCATER, V1.3 INVOKED BY:
LOC86 :F1:ADCINP.LNK TO :F1:ADCINP  MAP
 PRINT(:F1:ADCINP.MP2)&
BOOTSTRAP ORDER(CLASSES(DATA,STACK,CODE))&
ADDRESSES(CLASSES(CODE(0FC000H), DATA(000400H)))
WARNING 38: SEGMENT WITH MEMORY ATTRIBUTE NOT PLACED
 HIGHEST IN MEMORY SEGMENT: MEMORY
```

Figure 29. LOCATE Modification

The total control system for the application example can now be assembled using the hardware and software discussed in this note. The same "black box" approach used in hardware designs can be extended to include both software and hardware implementations. Figure 30 shows the complete solution to the control of up to eight agitated heating tanks.

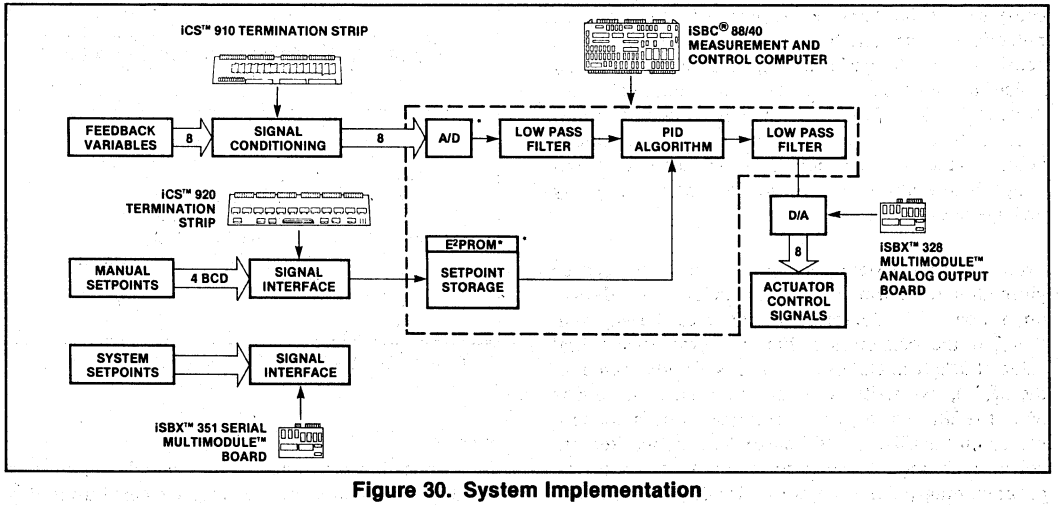

Figure 30. System Implementation

CONCLUSIONS

The purpose of this application note is to illustrate how the Intel iSBC 88/40 Measurement and Control Computer can be used to solve a complex control application. This has been done in Intel's lab and the results obtained from operating the board indicate that the system performance is sufficient to support the operation of eight loops each 100 milliseconds. Observed operation of the analog input and engineering unit conversion task indicated a 4 millisecond per channel execution time. The actual PID code required only 5 milliseconds per loop to execute.

The ease of implementation and fast execution time for multiple complex loops is a result of many Intel product features. For example, the use of the iRMX 88 Real Time Executive provided fast, small, and easy-to-use multitasking real time software for use on the single board computer. The iSBC 337 MULTIMODULE Numeric Data Processor Board enabled the use of high accuracy, easy-to-use floating point calculations without taking excessive execution time. If desired, a fixed point integer math algorithm could have been substituted for the floating point without changing the system performance appreciably. The iSBX 328 Analog Output MULTIMODULE Board provided low cost customization of the base board to support a variety of controllers and actuators.

Finally, the use of the Intel 2816 E^2PROM provided the non-volatile storage for system setpoints and constants which is required in a control situation.

Above all, the iSBC 88/40 Measurement and Control Computer provided a platform and execution vehicle for mounting and operation of the various ancillary features. Its iAPX 88/10 processor, memory and MULTIMODULE expansion sockets provided the flexibility to easily customize the board to a particular application environment.

APPENDIX A

APPLICATION CODE AVAILABILITY

The programs which were used to construct the application example are available from Intel through Insite. Insite, Intel's Software Index and Technology Exchange, is a collection of programs, subroutines, procedures and macros written by users of Intel's 8008, 8080, 8085, 8086, 8088, and 8048 microcomputers. Information on how to join Insite and obtain the source code can be obtained from your local Intel sales office or distributor, or by writing to:

North America

Intel Corporation
User's Library 6-5000
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

Europe

Intel International Corp. S.A.
User's Library
Rue du Moulin a Papier 51
Boite 1
B-1160 Brussels, Belgium

Orient

Intel Japan K.K.
User's Library
Flowerhill-Shinmachi, East Bldg.
1-23-9 Shinmachi, Setagaya-ku
Tokyo 154, Japan